# Problem 1 – Second Nature



Some unknown universal power has given you the idea to create a software program, which oversees the blooming of the legendary Edel flowers. The Edel flowers blooms when their Weiss dust is completely gone. Weiss dust is reduced by watering an Edel flower.

You will be given a **sequence of integers** – each indicating the Weiss dust in **each flower**. After that you will be given **another sequence of integers**. This time, each indicating – a bucket and the water in it.

Watering is done by picking **exactly one** bucket at a time. You must start picking from **the last received bucket** and start watering from **the first entered flower**. If the current bucket has **N** water, you **give** the **first entered flower** **N** water, **reducing** its Weiss dust by **N** – therefore **reducing** its integer value by **N**.

When an Edel flower’s **integer value** reaches **0 or less**, it blooms and **gets removed**. It is **possible** that the current flower’s value is **greater** than the current bucket’s value. **In that case** you **pick buckets** **until** you reduce its integer value to **0 or less**. If a bucket’s value is **greater** than the flower’s **current** value, you water the flower, and **add the remaining water** to the **next bucket in order**. In case there is no such, keep the remaining water in the same bucket.

If the **current bucket**’s value is **equal** to the **current flower**’s **current value**, the flower develops a **second nature**.   
By doing that, the flower becomes eternally bloomed – it cannot be affected by water. The bucket however **gets removed**.  
The watering is **continued** with the **next flower in order**.

If you **have managed** to **water all the flowers**, print the **remaining water buckets**, from the **last entered** **– to the first**.   
If you **haven’t managed** to water **all the flowers** with the given water, you must print the **remaining** **flowers**, by **order of entrance** – from the **first entered – to the last**.   
In both cases, you must also print the **second nature** flowers, if there are such, by **order of their appearance**.

### Input

* On the **first line** of input you will receive the integers, representing the **flowers**, and the Weiss dust in them – each with each, **separated** by a **single space**.
* On the **second line** of input you will receive the integers, representing the **buckets with water** – each with each, **separated** by a **single space**.

### Output

* On the first line of output you must print the remaining water buckets, or the remaining flowers, depending on the case you are in. Just **keep** the **orders of printing** exactly as **specified**.
* On the second line of output you must print the second nature flowers, by the order of their appearance.  
  If there are no second nature flowers, just print **“None”** on the second line of output.

### Constraints

* All of the given numbers will be valid integers in the range [1, 500].
* It is safe to assume that there will be **NO** case in which the water is **exactly as much** as the flowers’ values, so that at the end there are no flowers and no water.
* Allowed time/memory: 100ms/16MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| **4 2 10 5**  **3 5 5 11 6** | **9**  **None** | **We take the first entered flower, and the last entered bucket, as we are told by the description.**  **6 – 4 = 2 – we have 2 more so we add it to the next bucket in order and it becomes 13**  **13 – 2 = 11 -** again more, so we add it to 5 and it becomes 16  **16 – 10 = 6 – we add it to the next in order**  **11 – 5 = 6 – adds its value to the last bucket**  **We’ve managed to water all the flowers so we print what we have, remaining, from the water and the second nature flowers we found, in our case – None** |

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1 5 28 11 4**  **3 8 1 9 30 4 5** | **8 3**  **4** |